Towards the Temporal Streaming of Graph Data on Distributed Ledgers

Allan Third(✉), Ilaria Tiddi, Emanuele Bastianelli, Chris Valentine, and John Domingue

Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes MK7 6AA, UK
{allan.third,ilaria.tiddi,emanuele.bastianelli,chris.valentine,
john.domingue}@open.ac.uk

Abstract. We present our work-in-progress on handling temporal RDF graph data using the Ethereum distributed ledger. The motivation for this work are scenarios where multiple distributed consumers of streamed data may need or wish to verify that data has not been tampered with since it was generated – for example, if the data describes something which can be or has been sold, such as domestically-generated electricity. We describe a system in which temporal annotations, and information suitable to validate a given dataset, are stored on a distributed ledger, alongside the results of fixed SPARQL queries executed at the time of data storage. The model adopted implements a graph-based form of temporal RDF, in which time intervals are represented by named graphs corresponding to ledger entries. We conclude by discussing evaluation, what remains to be implemented, and future directions.

1 Introduction

This paper presents ongoing work in the use of distributed ledgers to provide validation for temporal graph-based data collected from sensor hardware. In particular, we use smart contracts executing on the Ethereum [16] distributed ledger to implement a named-graph-based temporal model for RDF data streams.

There are a number of motivating scenarios in which this can prove useful. [5] outlines criteria for the meaningful use of blockchain technologies. Among these are the requirements that data must be interacted with by multiple parties, who do not necessarily trust each other. We have identified two such scenarios among our current projects, relating to the collection of environmental data.

The GreenDATA project [11] focuses on gathering energy generation data from domestic generation systems. We collect this data from volunteers among colleagues, students, and other interested parties across the UK and elsewhere in order to make these datasets available to our students of sustainable energy modules, both to provide access to real system properties and also to encourage the development of data handling and analysis skills. One of the potential uses of distributed ledgers for renewable energy is to enable a disintermediated market: domestic producers could potentially sell surplus energy directly to domestic

E. Blomqvist et al. (Eds.): ESWC 2017 Satellite Events, LNCS 10577, pp. 327–332, 2017.
https://doi.org/10.1007/978-3-319-70407-4_41

consumers using cryptocurrency transactions on a blockchain. In such a scenario, with money changing hands, there would of course be a need for verification of data on behalf of both parties to a transaction. We therefore seek to allow students to model this scenario using a private distributed ledger in order to provide them with a way to explore how it might work.

The second scenario we are considering involves the collection of sensor data from moving vehicles, with limited network connectivity and computational power. Real-world situations where this might occur with a need for verification of data include the transport of environmentally-sensitive materials, such as food or medicines, disaster relief, or in long-distance motor racing, both of which can have financial or health consequences in the case of invalid data. We are planning to carry out a number of experiments in this setting over the next year, as two of the authors take part in the Mongol Rally [2] using a car equipped with a wide range of sensors and communication equipment to travel from Milton Keynes in the UK through Mongolia to Ulan Ude in Russia. The data gathered, with sub-second resolution on some sensors, and data gathered continuously while driving, will be streamed as RDF, network permitting, for on-the-fly and later analysis, including event detection. We intend to take this opportunity to experiment with the use of Ethereum light clients in a resource-limited setting and incorporating spatial data in our blockchain data handling.

2 Temporal Graphs and RDF Streams

The Resource Description Framework (RDF) [14] is a flexible semantic model for representing data, in the form of *triples* – "subject predicate object" sentences, with terms in each position represented by a (generally dereferencable) URL or, in the "object" position, a literal data value. One of the aims of RDF is to permit the easy linking and integration of data by means of URL matching and inference, for which use it has been highly successful [1]. The usual language for querying RDF data is SPARQL [13], although other approaches, such as Linked Data Fragments [12], are also used.

The typical use of RDF has been for the publication of datasets which are relatively static, with variability in the range of SPARQL queries used to extract information from them, severally or in combination. The "facts" represented by RDF triples are, in some sense, timeless, with issues about their lifespan or validity left outside the RDF model. In recent years, there has been increasing interest in the use of RDF to represent *streams* of temporally-annotated data, permitting explicit timestamps or time intervals to be associated with (sets of) RDF triples. This temporal aspect is essential for streaming data, as often the facts represented by the triples will only have limited temporal validity. It has been argued [15] that the approach to querying streams is the inverse to the usual querying model: highly volatile data with a small number of relatively static queries to extract information from them.

3 Distributed Ledgers and Smart Contracts

The Ethereum blockchain platform is a distributed ledger designed not just for cryptocurrency use but also as a decentraised computing platform. A *blockchain* is a data structure, duplicated on every node of a blockchain network, consisting of *blocks*, which are collections of *transactions* – records of transfers of cryptocurrency – between *accounts*. The creation of blocks – *mining* – is carried out by nodes, which compete for the opportunity to mine the next block at any given time. The choice as to which node may mine a block is made by consensus by some particular protocol, meaning that anyone seeking to insert a block containing an incorrect or fraudulent transaction must control or convince more than 50% of the nodes on the whole blockchain network to agree, and anyone seeking to alter an established transaction record must convince more than 50% of the nodes to roll back all transactions which have been recorded since the target record. In this way, in a large enough and diverse enough network, transactions on a blockchain can be trusted and effectively immutable. By encoding non-financial information in the transaction record, blockchains can be used to record trustworthy permanent records of other forms of data.

Ethereum specialises the blockchain concept further, by adding account types and addressing for *smart contracts*. A smart contract is a compiled unit of executable code which is stored on Ethereum and can be executed on all nodes via transactions involving the relevant account. As the compiled code is stored on the blockchain, it is possible to be assured that a particular contract has not been tampered with since it was compiled and deployed. In this way, Ethereum is intended to serve as a decentralised distributed computing platform.

Smart contracts have *state*, which can be updated by a contract when it is executed. The blockchain maintains a record of all previous states of a contract – inevitably, as to overwrite previous state would involve overwriting records earlier in the blockchain. Smart contracts can thus be used to implement a form of dynamic data storage with history in the Ethereum environment.

4 Use Cases in Detail

4.1 GreenDATA

The GreenDATA project [11] aims to collect data from domestic energy generation, from solar, wind and geothermal sources, with the purposes of making it available for students of sustainable energy, so that they might be able to study the behaviour of real systems in practice, in different locations and of different generation modalities. Contributors to GreenData have energy generating installations across the UK, and beyond, with participants in Austria and Crete.

Data is collected using either contributors' own hardware, or, more usually, using an OpenEnergyMonitor emonPi [8], a Raspberry Pi [9] based device which can be clamped to the appropriate cables of, for example, a solar photovoltaic system, and which then analyses the performance and behaviour of the system. The collected data can be stored locally to the emonPi, or, as in the GreenData

installations, transmitted over a domestic WiFi connection or a GSM signal to a remote data store. Types of data collected include grid supply voltage, power imported or exported, indoor and outdoor temperature, among others. The temporal resolution of the data is 10 s and data collection is continuous, 24 h a day. The timestamp of each data point is taken from GPS.

A modification to the emonPi software means that data is lifted to RDF on each device, before being sent to an RDF store hosted centrally, via an Ethereum light client [4] installed on the emonPi. The motivation behind having a blockchain infrastructure to verify the gathered data is twofold. Firstly, to serve as an experiment in the validation of data using blockchains in general, and secondly, to allow students to explore the potential role of blockchains in simulated disintermediated energy markets, in which consumer-producers of energy can trade energy surpluses directly with each other.

4.2 MK2MG – Milton Keynes to Mongolia

From mid-2017, two of the authors will be taking part in the Mongol Rally [2], driving an old car from Milton Keynes in the UK to a point near the border between Mongolia and Russia. The primary purpose of taking part is to raise money for charity, but we intend to use their journey to carry out a number of experiments using sensors attached to the car and both participants. Data relating to speed, location, temperature, humidity, heart rate and physical activity will be collected at a sub-second resolution, and both stored as RDF locally in on-car hardware and transmitted to a central server via a GSM connection. Event detection will be applied to the data in both locations. We aim to run an Ethereum light client on the in-car hardware in order to handle blockchain communications. In particular, we are interested in the results of streaming large quantities of data with blockchain-based recording in a scenario with connectivity and computational power limited by space, energy and cost.

The applications of the lessons we hope to learn from this exercise are in situations where there is a need for validated and trustworthy data in low power, intermittently connected settings, such as medication transport or disaster relief.

5 Temporal Graphs on the Ethereum Distributed Ledger

The Ethereum blockchain is not suited to storing large amounts of data – the speed of execution is unlikely to be fast enough to support high volume data streams. However, in order to achieve the goal of validation of data integrity, it is not necessary to store the data itself on the blockchain; all we need is to store sufficient metadata to allow anyone who does possess a chunk of the data to verify that its contents are intact. We need, therefore, a canonical representation of the data which can be hashed reproducibly to provide a verification – for example, the RDF serialisation of the source – and a reliable form of "punctuation" in the data stream, to identify complete chunks of data to be used for the hash.

The form of punctuation used depends on the temporal model used in the data. Multiple approaches have been taken to the representation of temporal

RDF streams. Broadly, they vary as to whether temporal information is associated with each triple individually ("triple-based") or with RDF graphs, where the triples in an individual graph share the same temporal information. In the latter approach, a graph usually corresponds to a time *interval*. In the former, the temporal information attached to a triple may either be an interval or a timestamp representing an instant. [3,6,7,10] The difference between interval and timestamp is not deeply relevant; with an interval representation, one can always simulate a timestamp by setting the start and end points of the interval to be identical, with no loss of information. For the purposes of this work, given the requirement to group sections of the data stream in order to implement what is needed for verification, it seems most appropriate to use the graph approach.

We therefore ensure that the data streams generated from sensors are segmented, at source, into named graphs corresponding to time intervals, with the length of intervals to be determined also at the source, and indicated within the data itself. Variable rather than static intervals provide more flexibility.

Smart contracts running on (a private instance of) the Ethereum blockchain have been written to receive the data, with each remote client provided with the address of the relevant contract(s). Each time data is sent, the contract identifies graphs specified by the source, and calculates a verification hash, extracting the start and end times of the interval covered by each graph. Four items – graph URI, hash, start and end time – are stored in the state of a new smart contract, the address of which is stored in a "master" contract and which, along with the original data, is forwarded onto a traditional RDF store.

At the same time, clients performing event detection construct an RDF representation of each event detected, and send it to a separate smart contract, along with the names and hashes of the relevant temporal graphs. The duplication of hashes provides a separate source of validity information for each graph.

In order to support the verification of data in standard SPARQL querying scenarios, a custom SPARQL endpoint, running off-blockchain, is being written to respond to federated SPARQL requests using the SERVICE keyword. Any triple patterns passed to this endpoint specified to be in a temporal graph known to be hashed on the blockchain are queried from the full dataset, and each relevant graph is hashed, and compared to the entries stored in the blockchain both from the streaming data contracts, and any relevant contracts from event detection. The custom endpoint returns a triple stating whether verification succeeded, allowing at least a base level of verification within SPARQL.

6 Conclusion and Future Plans

As stated at the outset, this paper presents a work-in-progress in the use of distributed ledger technology to provide a layer of verifiability to temporal RDF graphs containing streaming data. What we have achieved so far indicates that the approach proposed is practical to implement and flexible enough to cover the use cases proposed without limiting scope for further extension.

In practice, there are a number of parameters in this approach with which we can experiment to test their effects on performance, reliability and suitability

for the goals. These include, among others, the sizes of data batching, the use of on-chain vs. off-chain hashing and any compression approaches we can use with the data streams. Data we collect about the behaviour of, in particular, the MK2MG interactions with the blockchain will inform how best to handle limited connectivity and computational power systems.

In future work, we would like to explore the performance and execution cost implications of implementing at least some aspects of SPARQL directly inside smart contracts, to evaluate the possibility of having some limited temporal reasoning performed within a trusted context on the distributed ledger. We also intend to explore the use of distributed file systems to store the full data. It would be interesting, too, to explore how doing so might enable more Linked Data application scenarios to be implemented in a fully distributed, decentralised setting, with less reliance on external datastores as we do now, and following more closely the distributed ledger philosophy.

References

1. Abele, A., McCrae, J.P., Buitelaar, P., Jentzsch, A., Cyganiak, R.: Linking open data cloud diagram 2017 (2017). http://lod-cloud.net/
2. The Adventurists. Mongol Rally (2017). http://bit.ly/1gSXyjx
3. Bereta, K., Smeros, P., Koubarakis, M.: Representation and querying of valid time of triples in linked geospatial data. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 259–274. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38288-8_18
4. EthereumWiki. Ethereum light client protocol (2017). http://bit.ly/2qbOqmL
5. Greenspan, G.: Avoiding the pointless blockchain project, November 2015. http://bit.ly/2pfT43Z
6. Kietz, J.U., Scharrenbach, T., Fischer, L., Bernstein, A., Nguyen, K.: TEF-SPARQL: The DDIS query-language for time annotated event and fact triple-streams. Technical report, University of Zurich, Department of Informatics (2013)
7. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25073-6_24
8. OpenEnergyMonitor. OpenEnergyMonitor (2017). http://openenergymonitor.org
9. The Raspberry Pi Foundation. Raspberry pi (2017). http://raspberrypi.org
10. Tappolet, J., Bernstein, A.: Applied temporal RDF: efficient temporal querying of RDF data with SPARQL. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 308–322. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02121-3_25
11. Valentine, C.: GreenDATA (2016). http://projects.kmi.open.ac.uk/greendata
12. Verborgh, R., Vander Sande, M., Colpaert, P., Coppens, S., Mannens, E., Van de Walle, R.: Web-scale querying through Linked Data Fragments. In: LDOW (2014)
13. W3C. SPARQL (2008). https://www.w3.org/TR/rdf-sparql-query/
14. W3C. Resource Description Framework (2014). https://www.w3.org/RDF/
15. W3C. RDF stream models (2017). http://bit.ly/2pwWTFb
16. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper (2014)